Training Course on Advanced Embedded System Design with RTOS

Professional Development Training Course Outline

Category	Engineering	**Duration**	10 Days
Base Fee	$3,000 USD	**Delivery Mode**	Online / On-site Hybrid

Course Introduction

Introduction

Training Course on Advanced Embedded System Design with RTOS provides a comprehensive exploration of advanced embedded system design, with a particular focus on Real-Time Operating Systems (RTOS). Participants will delve into cutting-edge methodologies and best practices for developing high-performance, reliable, and secure embedded solutions. Through hands-on exercises and practical case studies, attendees will master critical concepts such as real-time scheduling, memory management, inter-task communication, and low-power design. This course is meticulously crafted to empower engineers with the specialized skills needed to navigate the complexities of modern embedded development, ensuring the creation of robust and efficient systems that meet the stringent demands of contemporary industrial and consumer applications.

The curriculum is engineered to bridge the gap between theoretical knowledge and practical application, emphasizing embedded Linux, ARM microcontrollers, IoT integration, and cybersecurity principles within an RTOS framework. We will explore advanced debugging techniques, performance optimization strategies, and the implementation of fault-tolerant designs. By the end of this program, participants will possess the expertise to architect and implement sophisticated embedded systems, leveraging the full potential of RTOS to deliver deterministic behavior and optimized resource utilization. This course is essential for professionals seeking to enhance their proficiency in a rapidly evolving field, enabling them to contribute to the next generation of intelligent and connected devices.

Programme Curriculum

Training Course on Advanced Embedded System Design with RTOS

Introduction

Training Course on Advanced Embedded System Design with RTOS provides a comprehensive exploration of advanced embedded system design, with a particular focus on Real-Time Operating Systems (RTOS). Participants will delve into cutting-edge methodologies and best practices for developing high-performance, reliable, and secure embedded solutions. Through hands-on exercises and practical case studies, attendees will master critical concepts such as real-time scheduling, memory management, inter-task communication, and low-power design. This course is meticulously crafted to empower engineers with the specialized skills needed to navigate the complexities of modern embedded development, ensuring the creation of robust and efficient systems that meet the stringent demands of contemporary industrial and consumer applications.

The curriculum is engineered to bridge the gap between theoretical knowledge and practical application, emphasizing embedded Linux, ARM microcontrollers, IoT integration, and cybersecurity principles within an RTOS framework. We will explore advanced debugging techniques, performance optimization strategies, and the implementation of fault-tolerant designs. By the end of this program, participants will possess the expertise to architect and implement sophisticated embedded systems, leveraging the full potential of RTOS to deliver deterministic behavior and optimized resource utilization. This course is essential for professionals seeking to enhance their proficiency in a rapidly evolving field, enabling them to contribute to the next generation of intelligent and connected devices.

Course duration

10 Days

Course Objectives

1. Master RTOS fundamentals and their application in mission-critical embedded systems.
2. Design and implement real-time scheduling algorithms for optimal system responsiveness.
3. Effectively manage memory resources in constrained embedded environments using advanced techniques.
4. Develop robust inter-process communication (IPC) mechanisms for complex embedded applications.
5. Apply power-efficient design strategies for extending battery life in IoT devices.
6. Integrate and configure embedded Linux distributions for specific hardware platforms.
7. Utilize ARM Cortex-M/R microcontrollers for high-performance embedded computing.
8. Implement secure embedded systems addressing current cybersecurity threats and vulnerabilities.
9. Employ advanced debugging and profiling tools for embedded software optimization.
10. Develop IoT-ready embedded solutions with cloud connectivity and data management.
11. Design and validate fault-tolerant embedded architectures for enhanced reliability.
12. Leverage machine learning at the edge within embedded constraints.
13. Apply DevOps principles to embedded software development workflows.

Organizational Benefits

1. Reduced development cycles due to enhanced team proficiency in RTOS and embedded design.
2. Improved product reliability and stability through the implementation of best practices in real-time systems.

3. Optimized resource utilization in embedded products, leading to cost savings and increased efficiency.
4. Enhanced system security and resilience against cyber threats, protecting intellectual property and user data.
5. Faster time-to-market for new embedded products by leveraging advanced design methodologies.
6. Development of innovative IoT solutions with robust connectivity and data processing capabilities.
7. Increased capacity for in-house embedded Linux development and customization.
8. Greater ability to design low-power, energy-efficient devices, meeting market demands.
9. Higher quality embedded software with fewer defects and reduced maintenance overhead.
10. Competitive advantage in the embedded systems market through the adoption of cutting-edge technologies.

Target Participants

- Embedded Systems Engineers
- IoT Developers
- Network Engineers
- Automation Engineers
- Data Scientists working with sensor data
- Application Developers
- R&D Engineers in Smart Systems
- Computer Science and Electrical Engineering Graduates

Course Outline

Module 1: RTOS Fundamentals and Core Concepts

- **Introduction to Real-Time Operating Systems (RTOS):** Definition, characteristics, advantages over bare-metal.
- **Tasks and Task Management:** Task states, priorities, task creation, deletion, and context switching.
- **Scheduling Algorithms:** Preemptive vs. non-preemptive, common algorithms (Round Robin, Rate Monotonic, EDF).
- **Interrupt Handling in RTOS:** ISR design, deferred interrupt processing, critical sections.
- **Case Study:** Designing a simple RTOS-based LED blinking application with multiple tasks.

Module 2: Inter-Task Communication and Synchronization

- **Semaphores and Mutexes:** Binary semaphores, counting semaphores, mutexes for resource protection.
- **Message Queues:** Asynchronous communication, message passing, producer-consumer patterns.
- **Event Flags and Mailboxes:** Signaling and data transfer mechanisms.
- **Deadlock Prevention and Priority Inversion:** Common issues and mitigation strategies.
- **Case Study:** Implementing a multi-threaded data acquisition system with sensor data exchange using message queues.

Module 3: Memory Management and Resource Optimization

- **Heap Management in RTOS:** Dynamic memory allocation, fragmentation issues, memory pools.
- **Memory Protection Units (MPU) and Memory Management Units (MMU):** Hardware-assisted memory control.
- **Stack Management and Overflow Detection:** Best practices for task stack sizing.
- **Low-Power Design Techniques:** Sleep modes, power gating, dynamic voltage and frequency scaling (DVFS).
- **Case Study:** Optimizing memory footprint for a battery-powered wearable device using memory pools and low-power modes.

Module 4: Embedded Linux and Advanced Architectures

- **Introduction to Embedded Linux:** Kernel architecture, user space vs. kernel space, build systems (Yocto, Buildroot).
- **Device Drivers Development:** Character devices, block devices, writing custom drivers.
- **Bootloader (U-Boot, GRUB) Customization:** Boot sequence, boot args, root filesystem.
- **Filesystems for Embedded Systems:** JFFS2, UBIFS, YAFFS2, choosing the right filesystem.
- **Case Study:** Porting a custom embedded Linux distribution to a new ARM development board.

Module 5: ARM Microcontroller Deep Dive

- **ARM Cortex-M Architecture:** Registers, instruction set, memory map, interrupt controller (NVIC).
- **Cortex-R and Cortex-A Overview:** Real-time and application processor architectures.
- **Peripheral Interfacing:** SPI, I2C, UART, ADC, DAC with direct register access and HAL libraries.
- **Debugging Techniques:** JTAG, SWD, trace capabilities, breakpoint management.
- **Case Study:** Developing a motor control application using a Cortex-M microcontroller with PWM and ADC.

Module 6: Embedded Cybersecurity

- **Threat Models for Embedded Systems:** Common vulnerabilities, attack vectors (physical, network, software).
- **Secure Boot and Firmware Updates:** Ensuring integrity and authenticity of software.
- **Encryption and Decryption:** Symmetric and asymmetric cryptography in resource-constrained environments.
- **Secure Communication Protocols:** TLS/SSL for IoT devices, secure data transmission.
- **Case Study:** Implementing secure over-the-air (OTA) updates for an IoT sensor node.

Module 7: IoT Integration and Cloud Connectivity

- **IoT Protocols:** MQTT, CoAP, HTTP for constrained devices.
- **Cloud Platform Integration:** AWS IoT, Azure IoT Hub, Google Cloud IoT Core.
- **Sensor Data Acquisition and Processing:** Interfacing various sensors, data formatting.
- **Edge Computing Concepts:** Processing data locally, reducing latency and bandwidth usage.
- **Case Study:** Building an end-to-end IoT solution for environmental monitoring, sending data to the cloud.

Module 8: Advanced Debugging and Profiling

- **Real-Time Trace and Event Logging:** Analyzing system behavior and performance.
- **Performance Counters and System Calls Tracing:** Identifying bottlenecks.
- **Memory Leak Detection and Analysis:** Tools and techniques for identifying memory issues.
- **Code Coverage and Unit Testing:** Ensuring software quality and robustness.
- **Case Study:** Profiling an RTOS application to identify CPU hotspots and optimize task execution times.

Module 9: Fault Tolerance and Reliability

- **Error Detection and Correction Codes (ECC):** Protecting memory integrity.
- **Watchdog Timers:** Recovering from software hangs and unexpected events.
- **Redundancy and Duplication:** Hardware and software redundancy strategies.
- **Self-Test and Diagnostics:** Built-in test capabilities for system health monitoring.
- **Case Study:** Designing a redundant control system for a critical industrial application.

Module 10: Machine Learning at the Edge

- **Introduction to TinyML:** Running ML models on resource-constrained devices.
- **Model Optimization Techniques:** Quantization, pruning, neural network accelerators.
- **Edge AI Frameworks:** TensorFlow Lite Micro, uTensor.
- **Data Collection and Preprocessing for Edge ML:** Preparing data for on-device inference.
- **Case Study:** Implementing a voice recognition model on a low-power microcontroller for a smart home device.

Module 11: Embedded Graphics and User Interfaces

- **GUI Frameworks for Embedded Systems:** LVGL, LittlevGL, Qt Embedded.
- **Display Technologies:** LCD, OLED, E-paper interfacing.
- **Touchscreen Integration:** Resistive and capacitive touch controllers.
- **Resource Optimization for Graphics:** Font management, image compression.
- **Case Study:** Developing a simple user interface for an embedded medical device with touch input.

Module 12: Advanced Driver Development

- **DMA (Direct Memory Access) Controllers:** High-speed data transfer without CPU intervention.
- **USB Device and Host Controllers:** Implementing USB functionality.
- **Networking Stacks (LwIP, µIP):** TCP/IP, UDP, network protocols.
- **Custom Peripheral Interfacing:** Designing drivers for unique hardware components.
- **Case Study:** Developing a custom USB mass storage device driver for an embedded system.

Module 13: Industrial Protocols and Connectivity

- **Modbus (RTU, TCP):** Industrial communication for SCADA and PLCs.
- **CAN Bus and CANopen:** Automotive and industrial automation networks.
- **EtherCAT and Profinet:** Real-time Ethernet for industrial control.
- **Fieldbus Technologies:** Overview of various industrial communication standards.
- **Case Study:** Integrating an embedded system into an industrial control network using Modbus TCP.

Module 14: Embedded System Verification and Validation

- **Hardware-in-the-Loop (HIL) Testing:** Testing embedded software with real hardware.

- **Software-in-the-Loop (SIL) Testing:** Simulating embedded software execution.
- **Test Automation Frameworks:** Automating embedded software testing.
- **Compliance and Certification:** Meeting industry standards and regulations.
- **Upcoming Scheduled Sessions**

Start Date	End Date	Location	Price
21 Sep 2026	02 Oct 2026	Online	$2,000
21 Sep 2026	02 Oct 2026	Physical	$3,000
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0
21 Sep 2026	02 Oct 2026	Physical	$0

Ready to enroll or request a customized program? Book online or contact us:

Register Online Now

Email: info@fineskilltrainingcenter.com | Website: www.fineskilltrainingcenter.com